



Python CoG Kit Tutorial

GPDK Workshop

Keith R. Jackson
Lawrence Berkeley National Lab
6/4/2001



Overview

- " Why Python?
- " Why a Python interface to Globus?
- " Architecture
- " Implementation / Design Issues
- " GramClient
- " IO
- " Grid FTP



Overview (cont.)

- " GASS
- " Security
- " To Do
- " Contacts



What's Python, and why use it?

- " It's an interpreted, object-oriented, high-level language with:
 - Automatic memory management
 - Built-in high-level data structures
 - Dynamic typing and binding
 - Excellent support for integrating with C/C++, Fortran, or Java (with JPython)
 - Simple, easy to learn syntax
 - A large collection of class libraries and modules



Why Python? (cont.)

- " Support for high-level constructs. (e.g., packages, modules, classes)
- " Support for generating and processing XML documents.
- " Platform independent GUI tools, and Open Source IDE's.
- " Runs on any platform with an ANSI-C compiler.
- " It's Open Source.



Why a Python interface to Globus?

- " Python offers support for high-performance scientific computing by providing:
 - Fast multi-dimensional array processing through Numerical Python (<http://www.pfdubois.com/numpy/>).
 - MPI access, and support for many popular computational libraries through Scientific Python (<http://starship.python.net/crew/hinsen/scientific.html>).
 - Excellent performance through the use of native extension modules.



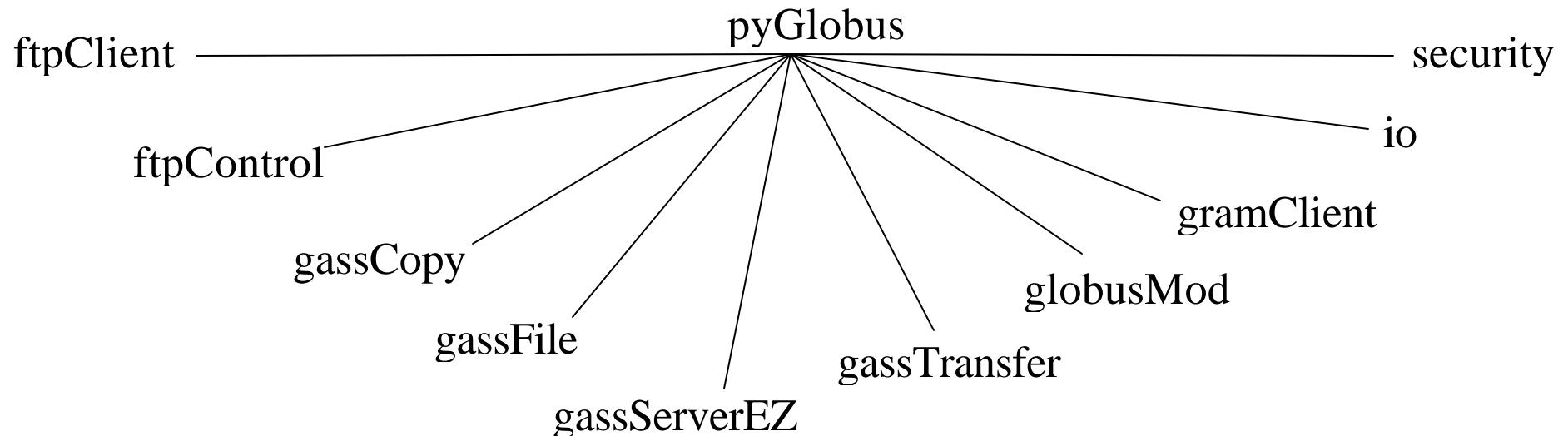
Why a Python interface to Globus? (cont.)

- " To support rapid development of high-level Grid services, while providing access to the underlying performance of the Globus toolkit.
- " To hide much of the complexity of Grid programming through the use of simple object-oriented interfaces.



Architecture

- “ A fairly direct mapping of the Globus architecture into Python, with a few adaptations to support the natural Python idiom.
- “ Contains the following package hierarchy:





Implementation / Design Issues

- " Implement the Python CoG kit as a Python extension module.
 - This provides very close to the native performance of Globus.
 - Allows easy adaptation to changes in Globus.
- " Use SWIG (<http://www.swig.org/>) to simplify generating the Python interface.
- " Attempt to map Globus to the most natural Python idiom.



GramClient

-
- " Provides an interface to the GRAM client library.
 - " Offers a single GramClient class:

try:

```
gramClient = GramClient.GramClient()
callbackContact = gramClient.set_callback(func, condV)
jobContact = gramClient.submit_request(rm, rsl,
                                        GramClient.JOB_STATE_ALL)
```

except GramClient.GramClientException, ex:

```
print ex.msg
sys.exit(-1)
```



IO

- " Provide access to high-performance IO through the Globus IO module.
- " Provides the following classes:



*Not yet completed.



IO (cont.)

TCP Server example:

```
attr = NetIOAttr.TCPIOAttr()
attr.set_authentication_mode(
    i o. GLOBUS_I O_SECURE_AUTHENTICATION_MODE_GSS_API)
authData = AuthData.AuthData()
authData.set_callback(auth_callback, None)
attr.set_authorization_mode(
    i o. GLOBUS_I O_SECURE_AUTHORIZATION_MODE_CALLBACK, authData)
attr.set_channel_mode(
    i o. GLOBUS_I O_SECURE_CHANNEL_MODE_GSI_WRAP)
soc = GSITCPSocket.GSITCPSocket()
port = soc.create_listener(attr)
soc.listen()
childSoc = soc.accept(attr)
buf = Buffer.Buffer(size)
bytesRead = childSoc.read(buf, size, size)
```



IO (cont.)

TCP Client example:

try:

```
attr = NetIOAttr.TCPI0Attr()
attr.set_authentication_mode(
    i o. GLOBUS_I0_SECURE_AUTHENTICATION_MODE_GSS_API)
authData = AuthData.AuthData()
attr.set_authorization_mode(
    i o. GLOBUS_I0_SECURE_AUTHORIZATION_MODE_SELF, authData)
attr.set_channel_mode(
    i o. GLOBUS_I0_SECURE_CHANNEL_MODE_GSI_WRAP)
soc = GSI_TCPSocket.GSI_TCPSocket()
soc.connect(host, port, attr)
nBytes = soc.write(str, len(str))
except globusException.GlobusException, ex:
    print ex.msg
    sys.exit(-1)
```



Grid FTP

- " Provides two modules: `ftpClient` and `ftpControl`.
 - The `ftpControl` module provides support for building Grid FTP servers, and low-level FTP control. It contains the following classes: `FtpControl`, `FtpControlLayout`, `FtpControlParallelism`, `FtpControlServer`, and `FtpControlTcpBuffer`.
 - The `ftpClient` module provides a simpler interface to the ftp client functionality, and provides the following classes: `FtpClient`, `FtpClientHandleAttr`, `FtpClientOperationAttr`, and `FtpClientRestartMarker`.



FtpClient example

try:

```
handl eAttr = FtpCl i entHandl eAttr. FtpCl i entHandl eAttr()
opAttr = FtpCl i entOperati onAttr. FtpCl i entOperati onAttr()
marker = FtpCl i entRestartMarker. FtpCl i entRestartMarker()
ftpCl i ent = FtpCl i ent(handl eAttr)
ftpCl i ent. get(url, opAttr, marker, done_func, condV)
handl e = ftpCl i ent. regi ster_read(buf, data_func, 0)
except globusExcepti on. GlobusExcepti on, ex
    print ex. msg
    sys. exit(- 1)
```



GASS

- " There are four modules wrapping the GASS functionality.
 - gassServerEZ: Provides a simple interface to creating a gassServer.
 - gassFile: Supports reading and writing remote GASS files as Python file objects.
 - gassCopy: Provides a uniform interface to reading remote files over a variety of protocols.
 - gassTransfer: Provides both client and server functionality to read and write remote files in a protocol independent way.



GAASS (cont.)

gassServerEZ example:

```
from pyGlobus.gassServerEZ import GassServerEZ  
  
try:  
    opts = LINE_BUFFER | STDOUT_ENABLE | STDERR_ENABLE  
    server = gassServerEZ(opts)  
    url = server.getURL()  
    . . .  
  
    del server  
except GassServerEZException, ex:  
    print ex.msg  
    sys.exit(-1)
```



GAASS (cont.)

gassFile example:

```
from pyGlobus.gassFile import GassFile  
try:  
    gFile = GassFile()  
    pyFile = gFile.fopen(url, "r")  
    for line in pyFile.readlines():  
        print line  
    gFile.close()  
except GassFileException, ex:  
    print ex.msg  
    sys.exit(-1)
```



GAASS (cont.)

gassCopy example:

```
from pyGlobus.gassCopy import GassCopyAttr
from pyGlobus.gassCopy import GassCopy

try:
    srcAttr = GassCopyAttr()
    handleAttr = GassCopyHandleAttr()
    destAttr = GassCopyAttr()
    ftpSrcAttr = FtpOperationAttr()
    ftpDestAttr = FtpOperationAttr()
    srcAttr.set_ftp(ftpSrcAttr)
    destAttr.set_ftp(ftpDestAttr)
    copy = GassCopy(handleAttr)
    copy.copy_url_to_url(srcUrl, srcAttr, destUrl, destAttr)
except GlobusException, ex:
```



Security

- " Provides low-level access to GSI functionality including:
 - Manipulating the grid-mapfile.
 - Creating and managing credentials.
 - Using the GSS-API.
 - Delegating credentials.



To Do

- " Finish wrapping the ftpControl, IO, and gassTransfer modules.
- " Wrap the new Replica Catalog functionality.
- " Develop higher-level components based on the current code to enable more rapid application development.
- " Develop a set of GUI components for job control, data transfer, etc.



Contacts

- " The code is available from:
<http://www-itg.lbl.gov/grid/projects/pyGlobus.html>
- " The mailing list, python-discuss@globus.org, is available for discussing issues related to the use of the Python CoG kit.
- " My email is: KRJackson@lbl.gov